FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 27, No 4, December 2014, pp. 521 - 542
DOI: 10.2298/FUEE1404521V

LOZENGE TILING CONSTRAINED CODES

Bane Vasić[1] and Anantha Raman Krishnan[2]
[1]Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721, USA
[2]Western Digital Corporation, Irvine, CA 92612, USA

Abstract: While the field of one-dimensional constrained codes is mature, with theoretical as well as practical aspects of code- and decoder-design being well-established, such a theoretical treatment of its two-dimensional (2D) counterpart is still unavailable. Research has been conducted on a few exemplar 2D constraints, e.g., the hard triangle model, run-length limited constraints on the square lattice, and 2D checkerboard constraints. Excluding these results, 2D constrained systems remain largely uncharacterized mathematically, with only loose bounds of capacities present. In this paper we present a lozenge constraint on a regular triangular lattice and derive Shannon noiseless capacity bounds. To estimate capacity of lozenge tiling we make use of the bijection between the counting of lozenge tiling and the counting of boxed plane partitions.
Keywords: 2D constrained codes, 2D constraints, lozenge tiling, colored tiling.

1 Introduction

Real-world communications channels, in contrast to channels of theoretical interest, often suffer from physical and systemic limitations that constrain the nature of data that may be transmitted through them. One such often-quoted example is the magnetic recording system, where information is translated to a direction of magnetization of the recording medium. Magnetic recording systems place constraints on the duration between two consecutive transitions of

Received August 9, 2014
Corresponding author: Bane Vasic
Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721, USA
(e-mail: vasic@ece.arizona.edu)

magnetization direction [1]. Transitions too close to each other degrades the signal-to-noise ratio (SNR) during readback. On the other hand, transitions too far apart make vulnerable to failure the timing-recovery sub-systems which typically rely on these transitions to derive symbol-level timing information. A solution to satisfy channel constraints is to use constrained codes (or modulation codes) – codes that transform user information to symbol-sequences that satisfy the constraints.

One dimensional (1D) constraints have been well-studied. The mathematical framework for determining the Shannon noiseless capacity of 1D constraints – the growth rate of the number of sequences satisfying a given constraint – has been comprehensively laid out. Further, development of 1D constrained codes and decoders is mature (refer [1] for a review on this topic). In contrast, analysis and code development for two-dimensional (2D) constraint systems have been less successful.

Initial work on 2D constraints was done by Ashley and Marcus [2] who considered 2D low-pass filtering codes for eliminating bit-patterns with large high-frequency components. Subsequently, considerable research was conducted on numerous classes of 2D constraints, e.g., [3–9]. In spite of these efforts, these constrained systems remain largely uncharacterized mathematically with only loose bounds of capacities existing. With an exception of a few cases, the channel capacity of 2D constrained channels is unknown (see [10]), and there is no systematic procedure to design encoders and decoders. Nonetheless, the need two-dimensional constrained codes has come to the fore in light of the recent work in two-dimensional magnetic recording (TDMR). In [11], it was shown that constrained coding which restricts the occurrence of certain 2D patterns greatly reduces system complexity and improves the detection performance. Also, there exist media models for TDMR that rely on polyomino tiling [12, 13], which are closely related to the 2D constrained systems.

In this paper, we consider a class of low-pass constraint on the triangular lattice termed *segregation* constraint and it special case, *no isolated bit* (NIB) constraint. We provide an upper bound on the Shannon noiseless capacity for this constraint. The key novelty of the our approach is to view our 2D constraint as a colored tiling of a plane. Thus, the estimation of the number of permissible colored tilings leads to upper bounds on Shannon noiseless capacity of 2D channels represented by this constraint. In addition, the tools used for analysis also lead to a framework for encoders and decoders for the constraint.

The rest of the paper is organized as follows. Section 2 gives the necessary background and motivation of the problem. In Section 3, we define lozenge codes and establish their connection with boxed plane partitions, and in Section 4 we derive bounds on capacity. Section 5 introduces encoding and decoding schemes for lozenge codes, and Section 6 concludes the paper.

2 Preliminaries

A two-dimensional constrained encoder may be visualized as a mapping from an unconstrained binary sequence into a colored tiling. A *tiling* of the plane is a collection of plane figures that fills the plane with no overlaps and no gaps. The plane figures used as building blocks for tilings are called *tiles*. Two tiles are said to be *neigbors* if they share an edge (side), and to be *touching* if they share only a single vertex. A *regular tiling* uses congruent regular polygons as tiles. There are only three regular tilings: triangular, square, and hexagonal. Figure 1 illustrates each of these tilings. A tiling of a plane figure or finite region can be defined analogously. A tiling is said to be *colored* or *labelled* if each of its tiles is asigned a color/symbol from a finite set of colors/symbols. A colored tiling is also referred as a *pattern* or a *configuration*. A binary coloring employs *black* and *white* tiles, while generally in M-ary coloring each tile is colored by one of $M > 1$ colors.

Consequent to the definitions above, a *two-dimensional constraint* can be expressed as a restriction on a coloring (or labeling) of tiles in a regular tiling. The most famous example is a hard-hexagon constraint [14] – a planar hexagonal lattice with nearest-neighbor exclusion. This is used as a gas model in statistical mechanics. The hard-hexagon constraint allows only those (binary) colorings in which black hexagons are isolated, i.e., have all white neighbors. A hard-triangle [3] and hard-square constraints are defined similarly. A two-dimensional *runlength* (d, k) constraint is a restriction on the separation space between black tiles, so that the number of white tiles between two black tiles in any direction is at least d and at most k $(0 \leq d < k)$. Indeed, it can be seen that the hard-hexagon, hard-triangle and hard-square constraints are instances of the (d, k) constraints. In particular, they are $(d, k) = (1, \infty)$ runlength constraints on their respective lattices. Constraints for which a coloring depends on both neigboring and touching tiles is referred to as a *checkerboard constraint* [9]. One example of a checkerboard constraint is a square tiling in which black squares are not permitted to share a vertex. As mentioned previously, both runlength and checkerboard constraints have been an active area of research, but the progress has been slow because of inherent hardness of two-dimensional tiling problems.

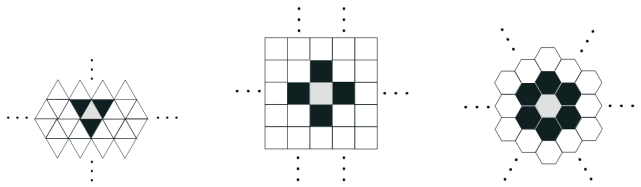
3 Lozenge Codes

Consider a regular tiling of a plane. A (d, k) *segregation* constraint is a tiling that limits the number of neighboring tiles (neighborhood size) of same color to be no less than $d+1$ and no more than $k+1$. A 2D bit pattern is said to satisfy a *no isolated bit* (NIB) constraint if every bit has at least one bit of the same polarity adjacent to it. Thus the NIB is a $(d, k) = (1, \infty)$ segregation constraint. Note that the parameters d and k in runlength and segregation constraints have different meanings. In rectangular latices runlength constraint can be converted

to a segregation constraint by *precoding* [1], while for other two latices in Fig. 1(a) this is not the case. In other words segregation constraints are not in bijection to simple runlength (d, k) constraints.

Using Fig. 1, the NIB constraint can be explained as follows: for any given tile (for example, the tile marked gray in the figure), if none of its neighbouring tiles (tiles marked black) are of the same colour, then the NIB constraint is said to be violated. Our recent experimental results show that if input bits are mapped to bit patterns satisfying the NIB constraint, then the probability of rewriting a grain is reduced [12, 15].

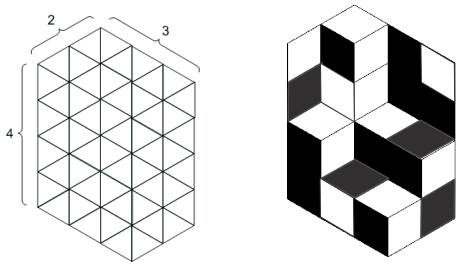
The goal of this paper is to establish a bound of Shannon noiseless capacity and designing of encoders and decoders for NIB constraint on the triangular lattice.



(a) (b) (c)

Fig. 1. Regular two-dimensional tilings (a) triangular, (b) square, and (c) hexagonal tiling. In the triangular, square and hexagonal tilings, each tile has three, four and six neighbors, respectively. The tiles shaded black are the neighbors that share an edge with the tile shaded gray.

To describe lozenge codes, we start by considering a hexagon $\mathcal{H}$ with sides of lengths a, b, c, a, b, c and angles of $2\pi/3$, subdivided into equilateral triangles of unit side by lines parallel to the hexagon sides. We will henceforth refer to such equi-angular triangularized hexagons as (a, b, c) hexagons. Figure 2(a) shows such a $(2, 3, 4)$ hexagon. As mentioned earlier, two triangles are neighbors if they share a side.



(a) (b)

Fig. 2. Lozenge tilings in a triangular lattice: (a) A $(2, 3, 4)$ hexagon $\mathcal{H}$ embedded in a triangular lattice. (b) Colored tiling of a (2,3,4) hexagon.

(a) (b) (c)

Fig. 3. The prototype tiles used in the tiling of hexagons.

We are interested in coloring the triangles using $M > 1$ different colors in such a way that no isolated triangle is colored differently than its neighbors. In the paper we focus on the $M = 2$ case. We require a segregation of at least two neighboring triangles of the same color. Fig. 2(b) shows one such coloring of the $(2, 3, 4)$ hexagon shown in Fig. 2(a).

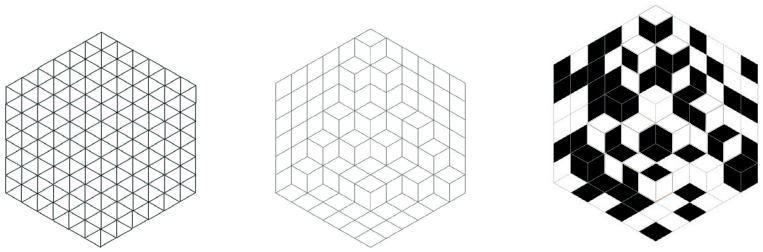
Two neighboring triangles form a rhombus with side of unit length and internal angles $\frac{\pi}{3}$ and $\frac{2\pi}{3}$. Such a rhombus is known as a *lozenge*. A lozenge created in this way may have three different orientations. Fig. 3 shows these orientations. We refer to the prototiles shown in Fig. 3(a), 3(b) and 3(c) as to type-A, type-B and type-C lozenges, respectively. The last two prototiles are referred to as vertical lozenges.

Now, suppose each of the lozenges is colored independently, we can ensure that for any triangle there is at least one neighboring triangle with the same color, i.e. $(d, k) = (1, \infty)$.

The constrained coding problem can be now formulated as follows: given a hexagonal region of a triangular lattice, one would like to find a one-to-one mapping from a set of integers representing user's binary data (this bijection is trivial) to a set of NIB-permissible colored tilings. In addition, due to complexity constrains, it is desirable that the mapping is described by an algorithm which operates locally on a small lattice region.

We propose a mapping which is a composition of two mappings: *tiling* and *coloring*. Now the NIB coloring can be separated into two steps as illustrated in Fig. 4. First, the hexagon (Fig. 4(a)) is tiled with (uncolored) lozenges (to obtain lozenge tiling in Fig. 4(b)), and then the lozenges are colored, (to obtain colored tiling in Fig. 4(c)). It is important to note that the reader does not have knowledge of the tiling and retrieves encoded bits solely based on the colored pattern. Thus, not all colorings can guaranty retrievability. Of interest are only those colorings that ensure retrievability.

Dividing the hard coloring hard problem into two manageable operations results into an unavoidable rate penalty, however the rate penalty due this simplification is a tradeoff for tractability. As we show in the following sections, it is now possible to use elegant combinatorial methods to design encoders and decoders. Moreover we can establish bounds on capacities, i.e., achievable density. We start with deriving the capacity bounds.



(a) (b) (c)

Fig. 4. Colored lozenge tilings in a triangular lattice: (a) A hexagon with side-length of 6 embedded in a triangular lattice. (b) An uncolored lozenge tiling of the hexagon. (c) A colored lozenge tiling of the hexagon.

4 Bounds on Capacity

For the NIB constraint defined above, we can define the Shannon noiseless capacity using an (n, n, n) hexagon as follows:

$$C = \lim_{n\to\infty} \frac{\log_2 M(n,n,n)}{log_2 2^{6n^2}} \quad (1)$$

$$= \lim_{n\to\infty} \frac{\log_2 M(n,n,n)}{6n^2}. \quad (2)$$

Note that since the number of triangles constituting the (n, n, n) hexagon is $6n^2$, in Eqn. 1, the denominator is logarithm of the total number of independent colorings of the constituent triangles. The numerator $M(n, n, n)$ is the logarithm of the number of colorings of the constituent triangles that satisfy the NIB constraint. Similarly, if $N(n, n, n)$ denotes the number of uncolored patterns (tilings), the asymptotic growth rate of $N(n, n, n)$ – we call it *tiling capacity* (C_T) is defined as

$$C_T = \lim_{n\to\infty} \frac{\log_2 N(n,n,n)}{log_2 2^{6n^2}}. \quad (3)$$

Finally, we define a coloring capacity (C_C) based on the number of distinct patterns, i.e. number $K(n, n, n)$ of ways of distinctly coloring a lozenge tiling so that the constraints are satisfied. This is defined as follows:

$$C_C = \lim_{n\to\infty} \frac{\log_2 K(n,n,n)}{log_2 2^{6n^2}}. \quad (4)$$

We are interested in calculating the number of colored lozenge tilings of a given lattice. Observe that this will yield bounds on the capacity of the NIB constraint because the capacity $C_{(1,\infty)}$ of the NIB constraint for the triangular lattice model, can be bounded by the tiling density and coloring densuty, C_T and C_C, respectively, as

$$C_{(1,\infty)} \geq C_T + C_C \quad (5)$$

To estimate capacity of lozenge tiling we make use of the bijection between the two enumeration problems: the counting of lozenge tiling and the counting of *boxed plane partition*. Then, we will make use of the work of MacMahon on calculating the number of boxed plane partition [16] to determine C_T. Before this, we briefly discuss the problem of enumeration of boxed plane partition.

The use of colored tilings to estimate capacity is attractive due to the fact that the theory of counting domino tilings is well-explored [17,18]. Early work in counting domino tilings includes the work of Kasteleyn [19], who calculated the number of domino tilings of a square lattice. Another work is that of MacMahon [16] in which the number of lozenge tilings of a hexagon embedded in a triangular lattice was calculated. Desreux and Remila [20] gave optimal algorithms for the generation of domino tilings and lozenge tilings.

4.1 A Method Based on Boxed Plane Partitions

A *plane partition* π is a collection of non-negative integers $\pi_{x,y}$ indexed by non-negative integers x, y such that (a) only finite number of $\pi_{x,y}$ are non-zero and, (b) $\forall x, y\ \pi_{x,y} \leq \pi_{x,y+1}$ and $\pi_{x,y} \leq \pi_{x+1,y}$. The plane partition is said to *fit inside* a box of dimension $a \times b \times c$ if there exist integers a, b, c such that $\pi_{x,y} \leq c$ for all x, y and $\pi_{x,y} = 0$ for all $x > a, y > b$. Such partitions are called *boxed plane partitions*. A more intuitive way of visualizing the boxed plane partitions is by constructing the *Young's solid diagram* corresponding to a boxed partition π. For instance, consider the following partition boxed within a box of dimension $2 \times 3 \times 4$.

$$\pi = \begin{pmatrix} 4 & 2 & 2 \\ 2 & 1 & 1 \end{pmatrix}$$

The dimension of the matrix is 2×3 with all entries ≤ 4. To build its corresponding Young's solid diagram, we first consider a box of dimension $2 \times 3 \times 4$. To one of the vertices, we assign the Cartesian co-ordinate $(0, 0, 0)$. To the opposite vertex we assign the co-ordinate $(2, 3, 4)$. For each $x \leq 2, y \leq 3$, we start at $(x - 1, y - 1, 0)$ and stack $\pi_{x,y}$ cubes of unit side length. This construction will fit inside the box.

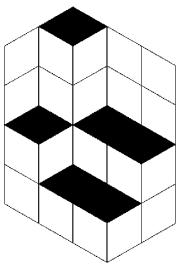


Fig. 5. Tiling corresponding to the boxed plane partition π.

The tiling given in Fig. 5 corresponds to the boxed plane partition π given

above. The Young's solid diagram construction is also apparent if the tiled $(2, 3, 4)$ hexagon is visualized as a $2 \times 3 \times 4$ box. For ease of visualization, the top face of the topmost box is colored black. The relationship between lozenge tiling and boxed plane partition is given by the Theorem 1 in [18].

Lemma 1 *The number of boxed plane partitions fitting inside a $a \times b \times c$ box is equal to the number of lozenge tilings of an (a, b, c) hexagon.*

Proof: The proof is based on MacMahon [16] formula.

4.2 Bounds on C_T

By Lemma 1, the number of lozenge tiling of any (a, b, c) hexagon is equal to the number of boxed partitions fitting inside an $a \times b \times c$ box. MacMahon [16] found the number of such partitions, $N_{(a,b,c)}$, to be $N_{(a,b,c)} = \prod_{i=1}^{a} (c+i)_b/(i)_b$ where $(i)_n := i(i+1)(i+2)\dots(i+n-1)$ is the *rising factorial*. For an equilateral hexagon, we were able to find the capacity of tiling (without coloring) in a closed form. It is given by Theorem 1.

Theorem 1 *The capacity of the lozenge tiling for an equilateral hexagon is $C_T = \frac{3}{4}\log_2 3 - 1$.*

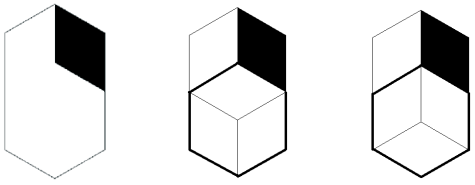
Proof: The proof follows from Lemma 1 as given in Appendix 7

Note that closed forms solutions for the capacity such as one given by the above theorem are quite rare in problems involving 2D constraints.

4.3 Bounds on C_C

The capacity of the coloring problem, C_C, can be estimated by counting the number of ways the lozenges constituting a lozenge tiling can be colored. Given a lozenge tiling of an (n, n, n) hexagon, $\mathcal{H}$, the coloring of the lozenges cannot be done arbitrarily. To explain this, we consider a colored lozenge tiling of the $(2, 1, 1)$ hexagon which is shown in Figure 6. Figure 6(a) shows the colored lozenge tiling without any boundaries. In the absence of the tiling information, the decoder would not be able to decode this tiling correctly. Figures 6(b) and 6(c) show two possible lozenge tilings this can be decoded to. This means that in the absence of knowledge of the tiling pattern, the colored tiling in Figure 6(a) is not uniquely decodable. In particular, the two tilings of the $(1, 1, 1)$ hexagon formed at the bottom of the larger hexagon (marked with bold edges) cannot be identified uniquely.

We denote the $(1, 1, 1)$ hexagon formed in Fig. 6(b) and Fig. 6(c) as type-1 and type-2 hexagon, respectively. To distinguish the two hexagons, we apply the following rule: Whenever a type-1 hexagon is encountered, the vertical lozenges of different orientation, i.e., type-B and type-C lozenges constituting



(a) (b) (c)

Fig. 6. Figure (a) shows a colored lozenge tiling for a $(2, 1, 1,)$ hexagon. In the absence of tiling information, the colored tiling in Figure (a) can be decoded to either of the colored tilings shown in Figures (b) and (c)

it are colored differently. One such coloring scheme is shown in Fig. 7, where type-B prototiles are always colored black, and type-C prototiles are always colored white. By using this coloring scheme, no information is stored in the vertical lozenges of the type-1 hexagon. We refer to this coloring scheme as *fixed-color vertical lozenges* (FCVL) coloring. The FCVL coloring constraint reduces the number of ways of coloring for each tiling. This reduction depends on the number of type-1 hexagons formed in a tiling. The capacity can be bounded by calculating the number of type-1 hexagons formed for every tiling.

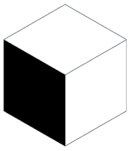


Fig. 7. The coloring scheme adopted for type-1 hexagon.

The coloring capacity, C_C, can be estimated by counting the number of ways the lozenges constituting a lozenge tiling can be colored. Given a lozenge tiling of an (n, n, n) hexagon, $\mathcal{H}$, the coloring of the lozenges cannot be done arbitrarily.

Theorem 2 *For the coloring of a lozenge tiling of an* (n, n, n) *hexagon* $\mathcal{H}$*, the capacity* $C_C \geq \frac{1}{3}$.

Proof: The proof is based on the equivalence between tilings and boxed plane partitions and Lemma 1 It is given in Appendix 8.

Note that the above bound on C_C may be strengthen at the expense of much harder combinatorial argument considering colors of tiles.

5 Encoding and Decoding Lozenge Codes

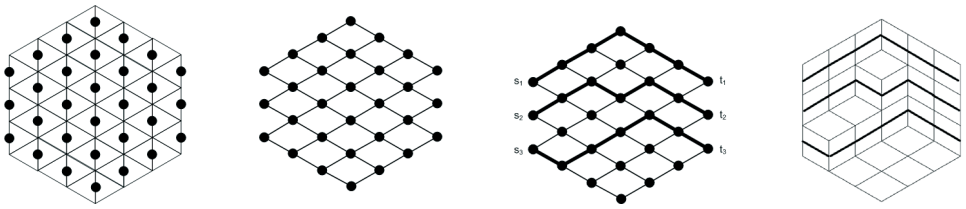
We refer to the assignment of a distinct element from the set of integers to an element of the set of all possible colored tilings as encoding of colored tilings. Again, we divide the problem into two separate problems: the encoding of tilings and the coloring of tiles. To facilitate this, we assume that the FCVL coloring constraint is imposed. The first step of the encoding process is the encoding of tilings for which we seek a bijective assignment between the set of integers and the set of all tilings. The second step is the coloring of the tiling. The second step is trivial since for the FCVL coloring, n^2 horizontal prototiles, which is the third of all prototiles in a (n, n, n) hexagon, can be colored independently. Thus we focus on the first step of an encoding algorithm.

Before discussing the proposed encoding algorithm we give an intuition behind our approach. We begin by recalling that there is a unique boxed plane partition corresponding to any lozenge tiling. Consider an (a, b, c) hexagon tiled with equilateral triangles with unit side-lengths. An alternative way to unambiguously specify a boxed plane partition is to traverse a paths covering only vertical prototiles as illustrated in Fig. 8(d) for the case of the $(3, 3, 3)$ hexagon. We refer to these paths as *routings*. Each section of the path corresponds to one type-B or type-C lozenge and it is drawn as a line segment of "NorthEast" or "SoutEast" orientation in Fig. 8(d). A collection of n such paths determines the tiling. Note, however, that the n paths in a collection cannot be chosen arbitrarily. This is simply because we do not allow that a box at any level "levitates" without being supported by boxes the beneath it. We describe this requirement though combinatorial objects known as *mountain ranges* illustrated in Fig. 8(c). Each path correspond to one mountain range, but "lower" paths cannot have mountain ranges higher than "upper" paths. We formalize this concept through lexicographical ordering and ranking of mountain ranges. A collection of n nondecreasing mountain ranges represent an instance of a tiling. The set of all such collections represent the set of all tilings. To perform encoding and decoding we need to enumerate mountain ranges and enumerate collection of mountain ranges. Thus the encoding procedure involves the step of converting an integer to a routing on a graph, which is represented by a set of constant-weight sequences and then converting it to a tiling. In the following subsections we introduce rigorously the combinatorial objects given in the above intuitive explanation. Before discussing the details of the proposed algorithm, we describe the problem of enumeration of the routings on a graph.

5.1 Routings on a Graph

Recall that the idea is to use the bijection between tiling and *routings on a graph*. Consider an (a, b, c) hexagon $\mathcal{H}$ tiled with equilateral triangles with unit side-lengths. For this tiling, an associated graph $\mathcal{G}$ can be constructed

as follows. The vertices of $\mathcal{G}$ are the mid-points of the vertical sides in the tiling of $\mathcal{H}$. Two vertices of $\mathcal{G}$ are connected if their corresponding sides lie on adjacent triangles. Fig. 8 shows a $(3,3,3)$ hexagon (Fig. 8(a)) and its corresponding graph (Fig. 8(b)). The leftmost nodes of $\mathcal{G}$ are designated as sources and the rightmost nodes as drains. Notice that the number of sources is equal to the number of drains. Traversing the sources (drains respectively) from top to bottom, each of the sources is denoted as $s_1, s_2, \ldots, s_c$ ($t_1, t_2, \ldots, t_c$ respectively). Now, the *routings* on the graph $\mathcal{G}$ is the set of c shortest non-intersecting paths from s_i to t_i for each i. The routings on the graph $\mathcal{G}$ are associated to the tilings of $\mathcal{H}$ with lozenges by the following theorem.



(a) (b) (c) (d)

Fig. 8. (a) A (3,3,3) hexagon $\mathcal{H}$ tiled with equilateral triangles of unit side. (b) The graph $\mathcal{G}$ associated with $\mathcal{H}$. (c) Example of a routing on the graph associated with a (3,3,3) hexagon. (d) The corresponding tiling on the hexagon.

Theorem 3 *The lozenge tilings on $\mathcal{H}$ corresponds bijectively to the set of routings on the associated graph $\mathcal{G}$.*

Proof: To prove this we make use of the Theorem 1 in [21].

Figures 8(c) and 8(d) show an example of the relationship between lozenge tiling on $\mathcal{H}$ and a given lattice routing on $\mathcal{G}$. Fig. 8(c) shows a routing on a graph associated with a $(3,3,3)$ hexagon (marked in bold). Fig. 8(d) shows the corresponding tiling on the hexagon. The routing overlaid on the hexagon illustrates how the tiling can be generated from the routing. It can be seen that each of the a paths of the routing has $a+b$ moves with b "Northeast (NE) moves" and c "Southeast (SE) moves" between nodes. By mapping the NE move to "1" and the SE move to "0", each routing can be represented as an ordered collection of *constant weight sequences*. Thus the problem reduces to coding of constant weight sequences. We now describe the encoding.

Let $\mathcal{R}$ be the ordered set of routings on the graph $\mathcal{G}$. The i^{th} element of $\mathcal{R}$, R_i, is an ordered collection of a constant weight sequences of length $a+b$ and weight b. We denote the ordered set of all sequences with length n and weight k as $\mathcal{D}_{(n,k)}$. Hence, each element of R_i is a member of $\mathcal{D}_{(a+b,b)}$. The encoding process consists of assigning to the number i, $0 \leq i \leq |\mathcal{R}|$, the i^{th} element of $\mathcal{R}$, R_i. In order to perform this operation without physically storing $\mathcal{R}$, algorithms which directly map an integer i to a distinct ordered

sets in $\mathcal{R}$ have been designed. To explain this, we consider the following: Let $R_i = \{d_i, i = 1, 2, \ldots, c | d_i \in \mathcal{D}_{(a+b,b)}\}$. There exists an ordering of elements of $\mathcal{D}_{(a+b,b)}$ such that $\forall i, j\ 1 \leq i, j \leq c$, if $i < j$, then $\mathsf{rank}(d_i) \leq \mathsf{rank}(d_j)$ where $\mathsf{rank}(.)$ denotes the rank of an element in the ordered set. Hence, instead of storing individual sequences, R_i can be represented using the ranks of d_i as $R_i = \{\mathsf{rank}(d_i), i = 1, 2, \ldots c |\ d_i \in \mathcal{D}_{(a+b,b)}\}$ (we will later show that this can be further simplified by the use of *non-decreasing strings*. This will be dealt with in detail in section 5.3). We refer to this as the rank-based representation. We now describe the *reverse lexicographical ordering* of constant weight sequences which is one such ordering of sequences.

5.2 Reverse Lexicographical Ordering of Constant Weight Sequences

In this section, methods to obtain a bijective mapping between members of $\mathcal{D}_{(n,k)}$ and integers will be developed. Firstly, we note that the size of $\mathcal{D}_{(n,k)}$, $\left|\mathcal{D}_{(n,k)}\right|$, is $\binom{n}{k}$. Also, note that each n-bit sequence has a decimal equivalent between 0 and $2^n - 1$. Let the decimal equivalent be denoted as $dE(.)$. The *lexicographical ordering* of the sequences is the arrangement of all the sequences in the ascending order of their decimal equivalent. The reverse lexicographical ordering can be defined analogously. In our description, we use the reverse lexicographical ordering. Table 1 shows the reverse lexicographical ordering for $\mathcal{D}_{(4,2)}$. The first column is the rank which gives the position of the sequence in the reverse lexicographically ordered set. Again, we use $\mathsf{rank}(.)$ to denote the rank of a particular element. It is easily seen that if $dE(A) \geq dE(B)$, then $\mathsf{rank}(A) \leq \mathsf{rank}(B)$.

Next, we consider the transformation of the members of $\mathcal{D}_{(n,k)}$, to paths in a Cartesian plane. We refer to these paths as *mountain ranges* (similar to the Catalan mountain ranges [22]). Let b be a member of $\mathcal{D}_{(n,k)}$. The mountain range $\mathsf{Mountain}(.)$ corresponding to b can be calculated as follows:

The mount ranges can also be represented by the n-tuple containing y coordinates of points traced by the sequence, denoted by $\mathbf{y}$. Figure 9 shows the

Table 1. Ordering of elements in $\mathcal{D}_{(4,2)}$.

Rank	String	Decimal equivalent
0	1100	12
1	1010	10
2	1001	9
3	0110	6
4	0101	5
5	0011	3

Algorithm 1 The Mount Range Algorithm

Start from $(x = 0, y = 0)$
for $i = 1$ to n **do**
  if $b(i) = 1$ **then**
    move to $(x + 1, y + 1)$
  else
    move to $(x + 1, y - 1)$
  end if
end for
Output the mount range Mountain(.)

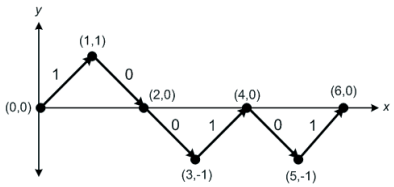


Fig. 9. Mountain range corresponding to the string 100101, Mountain(100101)

mountain range corresponding to the sequence 100101, a member of $\mathcal{D}_{(6,3)}$.

We now establish a relation between the lexicographical ranking and the *containment* of mountain ranges within one another. For any $A, B \in \mathcal{D}_{(n,k)}$, Mountain$(A)$ is said to be contained within $Mountain(B)$ (denoted as $Mountain(A) \subset Mountain(B)$) if each element of the n-tuple corresponding to A, $\mathbf{y_A}$ is never more than the corresponding element of the n-tuple corresponding to B, $\mathbf{y_B}$. For instance, the mountain range Mountain(1010) is contained in Mountain(1100). The following theorem relates the lexicographical ranking to containment.

Theorem 4 *For any $A, B \in \mathcal{D}_{(n,k)}$, if* Mountain$(A) \subset$ Mountain(B)*, then* rank$(A) \geq$ rank(B)

Proof: The proof is given in Appendix 9

The converse is not true. One counter-example in $\mathcal{D}_{(4,2)}$ is 1001 and 0110.

It is easily seen that an ordered set $\{d_i,\ i = 1, 2, \ldots, a | d_i \in \mathcal{D}_{(a+b,b)}\}$ such that Mountain$(d_i) \subset$ Mountain(d_{i+1}) for all i, is a member of $\mathcal{R}$ and hence corresponds to a valid tiling. This implies that the rank-based representation of R_i is a string of integers in which the integers are non-decreasing.

We now address ranking and unranking of members. Ranking, denoted by rank refers to the process mapping of a member of an ordered set to a distinct integer. Unranking is its inverse operation. We use ranking and unranking algorithms on the reverse lexicographically ordered set $\mathcal{D}_{(n,k)}$ for encoding data

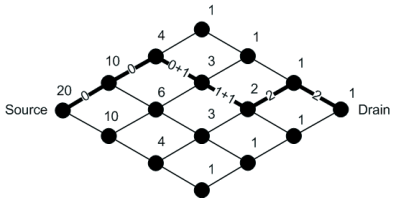


Fig. 10. Ranking of string 110010. Vertex labels denote the number of possible sequences from the corresponding node to a Drain. The edge labels on the mountain range, shown as a bold path, correspond to intermediate steps of rank computation.

to tilings. Ranking and unranking algorithms for this set are available [23, 24]. We have designed algorithms using methods similar to [22].

To perform the ranking, first the number of paths passing through every node (x, y) in $\mathcal{G}$, $n_{(x,y)}$ is calculated. The rank is calculated by starting with an initial estimate of rank $(= 0)$ and then refining the estimate as the sequence is parsed bit by bit. From any node (x, y), if the next bit is 0, the corresponding point in the Cartesian plane is $(x + 1, y - 1)$. This implies that all sequences which start from the node $(x+1, y+1)$ have a lower rank. Thus, given that the current estimate is e, the rank of this sequence is greater than $e + n_{(x+1,y+1)}$. Hence the estimate can be refined by adding the number to the current estimate. This process is continued till all the bits are parsed. The final estimate is the rank of the sequence.

Figure 10 shows flow of the algorithm for the sequence 110010, a member of $\mathcal{D}_{(6,3)}$. The number of sequences passing through every node is marked next to the node. The refinement of the rank at every bit is shown in the figure. At the last bit, the rank is calculated as 2.

Similar approach can be used to unrank an integer. Through the rest of this document, the ranking and unranking on this set will be denoted r_D and r_D^{-1} respectively.

It was earlier stated that R_i can be represented as $R_i = \{\mathsf{rank}(d_i), i = 1, 2, \ldots, c\}$. We note that the rank of each ds_i is between 0 and $\binom{a+b}{b}-1$. Also, we noted that for all d_i, d_j such that $i < j$, $\mathsf{rank}(d_i) \leq \mathsf{rank}(d_j)$. Now, we consider the rank-based representation which is the ordered set $\left\{\mathsf{rank}(d_i), i = 1, 2, \ldots, a | d_i \in \mathcal{D}_{(a+b,b)}\right\}$. This set can be expressed as a non-decreasing string of length c with members of the set $\left\{1, 2, \ldots, \binom{a+b}{b}\right\}$. We can further reduce the complexity of representation of elements of $\mathcal{R}$ if the non-decreasing strings can be ordered. Section 5.3 deals with this.

5.3 Lexicographical Ordering of Non-Decreasing Strings

This section deals with the lexicographical ordering in the set of *non-decreasing strings* of length n consisting of members of the set $\mathcal{N} = \{1, 2, 3, ..., N\}$. This

set is denoted $\mathcal{S}_n^N$ and is defined as the set of all strings, $\mathbf{d}$ of length n such that for all $i, 1 \leq i \leq n-1, d(i), d(i+1) \in \mathcal{N}$ and $d(i) \leq d(i+1)$. The set $\mathcal{S}_n^N$ is said to be lexicographically ordered if for any $d_1, d_2 \in \mathcal{S}_n^N, d_1(i) \leq d_2(i)\ \forall i \Leftrightarrow \mathsf{rank}(d_1) \leq \mathsf{rank}(d_2)$, where, as before, rank of a member is the position of the member in the ordered set. Table 2 shows the set $\mathcal{S}_3^2$ along with the rank of each element.

Table 2. Ordering of elements in $\mathcal{S}_3^2$.

Rank	String
0	111
1	112
2	122
3	222

To calculate the size of $\mathcal{S}_n^N$ we analyze the construction of strings of length k using strings of length $k-1$ for $k < n$. Consider a number i such that $1 < i \leq N$. The number of strings of length k ending in i, N_k^i is equal to the number of strings of length $(k-1)$ ending in digits less than or equal to i. That is,

$$\begin{aligned} N_k^i &= \sum_{j=1}^{i} N_{k-1}^j \qquad (N_1^i = 1, N_k^1 = 1) \\ &= \sum_{j=1}^{i-1} N_{k-1}^j + N_{k-1}^i \\ &= N_k^{i-1} + N_{k-1}^i \end{aligned} \tag{6}$$

This yields the recursive relation

$$N_k^i = N_k^{i-1} + N_{k-1}^i \qquad (N_1^i = 1, N_k^1 = 1) \tag{7}$$

The generating function for this recursion is given by $B_i(x) = \frac{x}{(1-x)^i}$. The coefficient of x^k in $B_i(x)$ gives the total number of sequences of length k ending in i. Since only the members of $\mathcal{S}_n^N$ can be used to make sequences of length $n+1$ ending in N, we have $\left|\mathcal{S}_n^N\right| = N_{n+1}^N$. By using the generating function, this is evaluated as $\binom{N+n-1}{n}$.

The ranking algorithm for the lexicographical ordering of members involves mapping an integer between 0 and $\binom{N+n-1}{n} - 1$ to a sequence in $\mathcal{S}_n^N$. As in the previous case, the ranking algorithm starts with an initial estimate ($e_0 = \binom{N+n-1}{n} - 1$) and refines its estimate as the string is parsed digit by digit. Given the estimate of the rank at the $(b-1)^{th}$ digit, e_{b-1}, and that the b^{th} digit is $d(b)$, the following conclusion can be drawn about the rank. The rank is less than or equal to $e_{b-1} - \binom{N'+b'-1}{b'}$ where $N' = N - d(b)$ and

$b' = n - b + 1$. N' gives the total number of members of $\mathcal{N}$ greater than $d(b)$ and b' gives the total number of available digits including the current digit. The second term in the refined estimate is the total number of strings of length b' that can be generated using the members of $\mathcal{N}$ larger than d. The strings having these substrings will have a higher ranking. Table 3 shows the progression of the algorithm for the member 112, a member of the set $\mathcal{S}_3^2$. At each step, the strings which are eliminated are shown in parentheses.

Table 3. Progression of the ranking algorithm for an example.

b	d_b	e_b
		3
1	1	$3 - 1 = 2$
		(222)
2	1	$2 - 1 = 1$
		(122)
3	2	1

The unranking of an integer can be achieved by using a similar approach. Through the rest of this document, the ranking and unranking on this set will be denoted r_S and r_S^{-1} respectively.

5.4 Encoding by Routing

Given the lattice $\mathcal{H}$ of dimension (a, b, c), the set of valid routings, $\mathcal{R}$, on $\mathcal{G}$ consists of ordered sets of size a consisting of members of the set $\mathcal{D}_{(a+b,b)}$. Also, the ordered sets will be in non-decreasing order of rank. Hence, these will correspond to a non-decreasing string of length a with members from the set $\{1, 2, 3, ..., \left|\mathcal{D}_{(a+b,b)}\right|\}$. This set is $\mathcal{S}_c^{\left|\mathcal{D}_{(a+b,b)}\right|}$. Hence, we have established a transitive relationship between $\mathcal{R}$ and $\mathcal{S}_c^{\left|\mathcal{D}_{(a+b,b)}\right|}$. However, all the sequences in this set do not correspond to valid tilings (as the converse of Theorem 4 is not true). Hence, we generate a look-up table (LUT) to map the integers in $[0, |\mathcal{R}| - 1]$ to the ranks of members of $\mathcal{S}_c^{D_{(a+b,b)}}$ corresponding to valid tilings. The encoding process is given by Algorithm 2.

The ranking algorithms, along with the LUT, can be used to map a tiling to a digit in $[0, |\mathcal{R}| - 1]$.

6 Conclusion

In this paper we first introduced a class of two dimensional constraints we call segregation constraints. In contrast to isolation runlength constraints considered in literature, we limit the minimal number of neighboring or touching tiles

Algorithm 2 The Encoding Algorithm

Use the lookup table (LUT) to find the rank of the corresponding string in $\mathcal{S}_c^{|\mathcal{D}_{(a+b,b)}|}$

Use the r_S^{-1} to find the string S_i in the set $\mathcal{S}_c^{D_{(a+b,b)}}$. Each digit in S_i, $S_i(j)$, corresponds to the rank of a sequence in $\mathcal{D}_{(a+b,b)}$.

for all $S_i(j)$ **do**

Use the r_D^{-1} to find the sequence corresponding to the rank $S_i(j)$.

end for

Assemble the sequences to find the ordered set in $\mathcal{R}$.

of the same color and restrict the shape of equaly-colored regions. For most two-dimensional recording systems such constraints are more natural than isolation constraints. The reason is that in high density recording systems recorded pattern features (areas of same physical properties) are smaller than what can be reliably distinguished by a reading device. For a particular constraint on a triangular lattice called the no-isolated bit constraint we introduced a method for encoding and decoding. Future work include studying the effect of lozenge constraints on a TDMR detector performance in the spirit of what we have done for the square lattice [25].

Acknowledgment

This work was supported by the National Science Foundation under Grant CCF-0963726 and CCF-1314147. The work of A. R. Krishnan was performed when he was with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, USA.
The authors would like to thank Seyed Mehrdad Khatami for his help and fruitful discussions.

7 Appendix

Proof:[Proof of Theorem 1] In this appendix we prove the Theorem 8. MacMahon formula [16] gives the number $N(a,b,c)$ of restricted plane partitions in the form

$$N(a,b,c) = \prod_{i=1}^{a}\prod_{j=1}^{b}\prod_{k=1}^{c}\frac{i+j+k-1}{i+j+k-2}$$

It can be rewritten as

$$N(a,b,c) = \prod_{i=1}^{a}\frac{(c+i)_b}{(i)_b}$$

where $(i)_n := i(i+1)(i+2)...(i+n-1)$ is the *rising factorial*.

Since $(i)_n = (i+n-1)!/(a-1)!$, we have

$$\prod_{i=1}^{a} \frac{(c+i)_b}{(i)_b} = \prod_{i=1}^{a} \frac{(c+i+b-1)!(i-1)!}{(c+i-1)!(i+b-1)!}$$

Each product of factorials in 7 is of the form $\prod_{i=1}^{a} (d+i-1)!$ and can be written as

$$\prod_{i=1}^{a}(d+i-1)! = \frac{\prod_{i=1}^{d+a-1} i!}{\prod_{i=1}^{d-1} i!}$$

The superfactorials $\prod_{i=1}^{a} i!$ in 7 can be expressed as $\prod_{i=1}^{n-1} i! = G(n+1)$. Therefore we obtain

$$N(a,b,c) = \prod_{i=1}^{a} \frac{G(a+b+c+1)G(a+1)G(b+1)G(c+1)}{G(a+b+1)G(a+c+1)G(b+c+1)} \tag{8}$$

where in Eq. 8 G denotes the Barnes G-Function [26] defined by

$$G(d+1) = (2\pi)^{\frac{d}{2}} e^{-\frac{1}{2}\left(d(d+1)+\gamma d^2\right)} \cdot \\ \cdot \prod_{i=1}^{+\infty} \left(\left(1+\frac{d}{i}\right)^{i} e^{-d+d^2/(2i)} \right)$$

For a regular hexagon with $a = b = c = n$ we have

$$N(n,n,n) = \prod_{i=1}^{a} \frac{G(3n+1)\left(G(n+1)\right)^3}{G(2n+1)^3} \tag{9}$$

Using the asymptotic of $G(d+1)$

$$\ln G(d+1) \tilde{} z^2(\frac{1}{2}\ln z - \frac{3}{4}) + \frac{1}{2}\ln(2\pi)z - \\ -\frac{1}{12}\ln z + \xi^{'}(-1)O(\frac{1}{2})$$

we obtain

$$\begin{aligned} \ln N(n,n,n) &\sim \ln G(3n+1) + 3\ln G(n+1) - \\ &\quad -3\ln G(2n+1) \\ &\sim n^2\left(\frac{9}{2}\ln 3 - 6\ln 2\right) + \\ &\quad +n\ln 2\pi - \frac{1}{12}\ln n - \frac{1}{12}\ln\frac{3}{2} \end{aligned}$$

By changing the base of the logarithm we obtain

$$\log_2 N(n,n,n) \sim n^2\left(\frac{9}{2}\log_2 3 - 6\right) \tag{10}$$

The area of the hexagon with sides a, b and c is $A(a,b,c) = 2(ab+ac+bc)\frac{\sqrt{3}}{4}$. For a regular hexagon $A(n,n,n) = \frac{2\sqrt{3}}{2}n^2$, so that finally the density is

$$\begin{aligned} D &= \lim_{n\to\infty} \frac{\log_2 N(n,n,n)}{A(n,n,n)} \\ &= \lim_{n\to\infty} \frac{n^2\left(\frac{9}{2}\log_2 3 - 6\right)}{\frac{2\sqrt{3}}{2}n^2} \\ &= \sqrt{3}(\log_2 3 - \frac{4}{3}) \end{aligned}$$

8

Proof:[Proof of Theorem 2] Consider the equivalence between tilings and boxed plane partitions. It can be seen that each box in the Young's solid diagram of an (a,b,c) hexagon with 3 visible faces corresponds to a type-1 hexagon. To find the maximum number of type-1 hexagon in $\mathcal{H}$, the Young's solid diagram corresponding to an $n \times n \times n$ box with the maximum number of visible boxes is constructed. This can be constructed as follows: Start at $(x,y) = (0,0)$ of the $n \times n \times n$ box and stack n boxes. Now, at $(x,y) = (1,0)$, stack $n-1$ boxes. Step in the x direction reducing the number of boxes stacked by 1 at each step. This is continued till $(x,y) = (n-1,0)$ where 1 box is stacked. Next, start at $(x,y) = (0,1)$ and stack $n-1$ boxes. Continue the same process till $(x,y) = (n-2,1)$ is reached where 1 box is stacked. This process is continued till $(x,y) = (n-1,0)$ is reached where 1 box is stacked. This corresponds to the Young's solid diagram with the maximum number of visible boxes. The boxed plane partition corresponding to this Young's diagram is given by:

$$\pi = \begin{pmatrix} n & n-1 & \dots & 3 & 2 & 1 \\ n-1 & n-2 & \dots & 2 & 1 & 0 \\ n-2 & n-3 & \dots & 1 & 0 & 0 \\ \vdots & \vdots & & & & \\ 1 & 0 & \dots & & & 0 \end{pmatrix}$$

As an example, Figure 11 shows the lozenge tiling of a $(4,4,4)$ hexagon with the maximum number of type-1 hexagons. Alternatively, it can be visualized as a the Young's solid diagram corresponding to a $4 \times 4 \times 4$ box.

The number of visible boxes is equal to the number of non-zero entries in π. This is equal to $\frac{n(n+1)}{2}$. This is equal to the number of type-1 hexagons formed in $\mathcal{H}$. The number of lozenges where no information is stored is twice the number of such hexagons. By subtracting this from the total number of lozenges in $\mathcal{H}$, the number of bits that can be stored in this region can be found.

The total number of lozenges is half the number of equilateral triangles tiling $\mathcal{H}$. By calculating the areas of $\mathcal{H}$ and a unit triangle, the number of

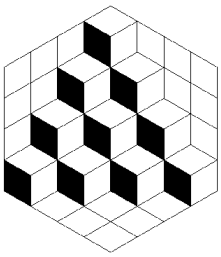


Fig. 11. The lozenge tiling of a $(4, 4, 4)$ hexagon with maximum number of type-1 hexagons. It can also be visualized as the Young's solid diagram of a $4 \times 4 \times 4$ box with the maximum number of visible boxes.

triangles is calculated as $6n^2$. Hence, the number of lozenges is $3n^2$. Since for any lozenge tiling of $\mathcal{H}$, the number of type-1 hexagons is always less than or equal to $n(n+1)$, we have a lower bound on the capacity C_C as follows:

$$C_C \quad \geq \quad \lim_{n\to\infty} \frac{3n^2 - n(n+1)}{6n^2} = \frac{1}{3} \tag{11}$$

9

Proof:[Proof of Theorem 3] If $\mathsf{Mountain}(A) \subset \mathsf{Mountain}(B)$, then $y_A(i) \leq y_B(i)$ for all i. It is enough to prove that if $y_A(i) \leq y_B(i)\ \forall\ i = 1, 2, \ldots, n$, then $A \leq B$. In order to do this, reconstruct the sequences A and B from $\mathbf{y_A}$ and $\mathbf{y_B}$. Let $t, 1 \leq t \leq n$ such that $\forall i < t, y_A(i) = y_B(i)$ and $y_A(t) \neq y_B(t)$. This means that the i^{th} term of A is 0 and that of B is 1 (to satisfy the inequality in the terms of $\mathbf{y_A}$ and $\mathbf{y_B}$). Hence, $dE(A) \leq dE(B)$ (inequality occurs if they differ in one or more bits and equality occurs if they do not differ). Hence $\mathsf{rank}(A) \geq \mathsf{rank}(B)/$

References

[1] B. H. Marcus, P. H. Siegel, and J. K. Wolf, "Finite-state modulation codes for data storage," *IEEE Journal of Selected Areas in Communication*, vol. 10, no. 1, pp. 5–37, 1992.

[2] J. J. Ashley and B. H. Marcus, "Two-dimensional low-pass filtering codes," *IEEE Transactions on Communications*, vol. 46, pp. 724–727, Jun 1998.

[3] Z. Nagy and K. Zeger, "Capacity bounds for the hard-triangle model," *Proceedings of International Symposium on Information Theory 2004*, p. 162, 2004.

[4] I. Demirkan and J. K. Wolf, "Block codes for the hard-square model," *IEEE transactions on Information Theory*, vol. 51, no. 8, p. 2836, Aug 2005.

[5] A. Kato and K. Zeger, "Partial characterization of the positive capacity region of the two-dimensional asymmetric run length constrained channels," *IEEE Transactions on Information Theory*, vol. 46, no. 7, p. 2666, Nov 2000.

[6] ——, "On the capacity of two-dimensional run-length constrained channels," *IEEE Transactions on Information Theory*, vol. 45, no. 5, p. 1527, July 1999.

[7] Z. Nagy and K. Zeger, "Bit-stuffing algorithms and analysis for run-length constrained channels in two and three dimensions," *IEEE Transactions on Information Theory*, vol. 50, no. 12, p. 3146, Dec. 2004.

[8] P. Siegel and J. Wolf, "Bit-stuffing bounds on the capacity of 2-dimensional constrained arrays," in *International Symposium on Information Theory*, August 16 - August 21 1998, p. 323.

[9] Z. Nagy and K. Zeger, "Asymptotic capacity of two-dimensional channels with checkerboard constraints," *IEEE Transactions on Information Theory*, vol. 49, no. 9, p. 2115, 2003.

[10] T. Etzion and K. G. Paterson, "Zero/positive capacities of two-dimensional runlength constrained arrays," in *In Proc. 2001 IEEE Intl. Symp. on Inform. Theory*, 2004, p. 269.

[11] S. Khatami and B. Vasic, "Generalized belief propagation detector for TDMR microcell model," *IEEE Transactions on Magnetics*, vol. 45, no. 10, p. submitted, October 2012.

[12] A. Krishnan, R. Radhakrishnan, B. Vasic, A. Kavcic, W. Ryan, and F. Erden, "2-d magnetic recording: Read channel modeling and detection," *IEEE Trans. on Magn.*, vol. 45, no. 10, pp. 3830 –3836, Oct. 2009.

[13] L. Pan, W. E. Ryan, R. Wood, and B. Vasic, "Coding and detection for rectangular-grain TDMR models," *IEEE Trans. Magn.*, vol. 47, no. 6, pp. 1705–1711, Jun. 2011.

[14] R. J. Baxter, "Hard hexagons: exact solution," *Journal of Physics A: Mathematical and General*, vol. 13, no. 3, p. L61, 1980. [Online]. Available: http://stacks.iop.org/0305-4470/13/i=3/a=007

[15] A. Krishnan, R. Radhakrishnan, and B. Vasić, "Read channel modeling for detection in two-dimensional magnetic recording systems," *IEEE Transactions on Magnetics*, vol. 45, no. 10, pp. 3679–3682, October 2009.

[16] P. A. MacMahon, *Combinatory Analysis (Volume II), reprint.* New York: Chelsea Publishing Company, 1990.

[17] H. Cohn, M. Larsen, and J. Propp, "The shape of a typical boxed plane partition," *New York Journal of Mathematics*, vol. 4, p. 137, 1998. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:math/9801059

[18] G. David and C. Tomei, "The problem of calissons," *American Mathematics Monthly*, vol. 96, no. 5, pp. 429–431, 1989.

[19] P. W. Kasteleyn, "The statistics of dimers on a lattice: The number of dimer arrangements on a quadratic lattice," *Physica*, vol. 12, pp. 1209–1225, 1961.

[20] S. Desreux and E. Remila, "An optimal algorithm to generate tilings," *Journal of Discrete Algorithms*, vol. 4, no. 1, pp. 168 – 180, 2006.

[21] M. Luby, D. Randall, and A. Sinclair, "Markov chain algorithms for planar lattice structures (extended abstract)," *IEEE Symposium on Foundations of Computer Science*, pp. 150–159, 1995. [Online]. Available: citeseer.ist.psu.edu/luby95markov.html

[22] D. L. Kreher and D. R. Stinson, *Combinatorial Algorithms: Generation, Enumeration and Search.* CRC Press, 1999, pp. 97–101.

[23] B. Ryabko, "Fast enumeration of combinatorial objects," *Math. and Applications*, vol. 10, p. n2, 1998. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0601069

[24] T. M. Cover, "Enumerative source coding," *IEEE transactions on Information Theory*, vol. IT-19, no. 1, p. 73, Jan 1973.

[25] M. Khatami and B. Vasic, "Constrained coding and detection for TDMR using generalized belief propagation," in *Proc. IEEE Int. Comm. Conf.*, Sydney, Australia, Jun. 10–14 2014.

[26] E. W. Barnes, "The theory of the G-Function," *Quart. J. Pure Appl. Math*, vol. 31, pp. 264–314, 1900.